DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

ACCESS TO ALL CLOUD SERVICES WITH A SINGLE SIGN-ON (SSO) MOBILE APPLICATION

Ramazan AKMAN

December, 2025

İZMİR

ACCESS TO ALL CLOUD SERVICES WITH A SINGLE SIGN-ON (SSO) MOBILE APPLICATION

A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for Master of Science in
Department of Computer Engineering Program

by

Ramazan AKMAN

December, 2025

İZMİR

M.ScTHESIS EXAMINATION RESULT FORM

We have read the thesis entitled**"ACCESS TO ALL CLOUD SERVICES WITH A SINGLE SIGN-ON (SSO) MOBILE APPLICATION"**completed by **RAMAZAN AKMAN** under supervision of **PROF. DR. GÖKHAN DALKILIÇ** and we certify that inour opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Gökhan DALKILIÇ

Supervisor

Prof. Dr. Mehmet Hilal ÖZCANHAN

Jury Member

Doç. Dr. Ömer AYDIN

Jury Member

Prof. Dr. Abdullah SEÇGİN

Director

Graduate School of Natural and Applied Sciences

ACKNOWLEDGMENT

I would like to express my deepest gratitude to Prof. Gökhan Dalkılıç for his valuable guidance, insightful feedback, and continuous support throughout the preparation of this thesis. His expertise, encouragement, and patience have been invaluable to the completion of my work. I am also thankful to all my colleagues, friends, and family members who have supported me during this journey.

Ramazan AKMAN

ACCESS TO ALL CLOUD SERVICES WITH A SINGLE SIGN-ON (SSO) MOBILE APPLICATION

ABSTRACT

The rapid adoption of microservice-oriented architectures and cloud-based systems has intensified the need for seamless authentication across domains and subdomains. Traditional Single Sign On (SSO) which depends on browser redirection and password entry, often results in poor mobile usability and increased phishing risks. This thesis introduces the Mobile Single Sign On (MSSO) system, a domain-agnostic and multi-tenant architecture designed to unify authentication across affiliated applications. MSSO leverages QR-code-based and truly passwordless authentication, supported by a secure STOMP/WebSocket real-time token delivery. The authentication flow consists of Domain Binding—where the mobile application authenticates with the domain's Identity Provider to obtain a long-lived Master Token—and a rapid passwordless Application Login stage. A lightweight JavaScript SDK embedded in web applications generates a session-bound QR code and opens a real-time listener, while the MSSO Mobile application uses the Master Token to authorize the login request and deliver a short-lived JSON Web Token (JWT) directly to the browser.

A comprehensive security analysis evaluates risks specific to QR-based real-time exchange, including QR hijacking, token replay, session binding attacks, and Master Token protection. Performance testing validates system scalability under concurrent load. Overall, the findings confirm that MSSO provides a secure, efficient, and user-centric alternative for identity management in modern multi-domain enterprise environments.

Keywords: Single Sign-On, mobile authentication, passwordless security

ACCESS TO ALL CLOUD SERVICES WITH A SINGLE SIGN-ON (SSO) MOBILE APPLICATION

ÖZ

Mikroservis odaklı mimarilerin ve bulut tabanlı sistemlerin hızla benimsenmesi, farklı alanlar ve alt alanlar arasında kesintisiz kimlik doğrulama ihtiyacını artırmıştır. Geleneksel Tek Oturum (SSO) yapıları, tarayıcı yönlendirmesine ve parola girişine bağlı oldukları için genellikle oltalama (phishing) riskleri doğurur. Bu tez, ilişkili uygulamalar arasında kimlik doğrulamayı birleştirmek üzere tasarlanmış, alan bağımsız bir mimari olan Mobil Tek Oturum (MSSO) sistemini tanıtmaktadır. MSSO, QR kod tabanlı ve tamamen parolasız kimlik doğrulamayı kullanmaktadır. Kimlik doğrulama akışı, mobil uygulamanın kimlik sağlayıcısı ile doğrulama yaparak uzun ömürlü bir anahtar alması ile başlar ve ardından hızlı, parolasız uygulama girişi için kod tarama ekranına geçilir. İstemci web uygulamalarına gömülü kod parkçacıkları, oturuma bağlı bir kod üretir ve gerçek zamanlı bir dinleyici ile dinleme başlatılır. Mobile uygulama kodu tarar ve kayıtlı anahtarı kullanarak giriş isteğini oturum sunucusuna iletir ve kısa ömürlü bir anahtar JSON Web Token (JWT) formatında üretilir ve soket aracılığıyla istemciye iletir.

Bu tür bir mimaride ortaya çıkabilecek güvenlik riskleri kapsamlı bir şekilde değerlendirilmiştir. QR kod hırsızlığı, kısa sureli anahtarın yeniden kullanımı, uzun sureli anahtarın ele geçirilmesi gibi riskler dikkate alınarak güvenlik önlemleri alınmıştır. Gerçekleştirilen testlerde sistemin eşzamanlı yük altında ölçeklenebilirliği incelenmiştir. Genel olarak bulgular, MSSO'nun modern çok alanlı kurumsal ortamlarda güvenli, verimli ve kullanıcı odaklı bir kimlik yönetimi alternatifi sunduğunu ortaya koymaktadır.

Anahtar kelimeler: Single Sign-On, mobil kimlik doğrulama, parolasız giriş

CONTENTS

Page

M.ScTHESIS EXAMINATION RESULT FORM ii
ACKNOWLEDGMENT iii
ABSTRACT iv
ÖZ v
CONTENTS vi
LIST OF FIGURES ix
LIST OF TABLES x

CHAPTER ONE -INTRODUCTION 1

1.1 Study Area 1
1.2 Purpose & Scope 2
1.3 Motivation 3
1.4 Research Questions 3
1.5 Contributions 4
1.6 Outline Of The Report 5
1.7 Scope 6

CHAPTER TWO -PREVIOUS STUDIES 8

2.1 Overview of Single Sign-On Architectures 8
2.2 Single Sign-On Classifications 8
2.2.1 Local Pseudo SSO 8
2.2.2 Proxy-Based Pseudo SSO 9
2.2.3 Local True SSO 9
2.2.4 Proxy-Based True SSO 10
2.2.5 Comparison of SSO Architecture Types 10
2.3 Modern Passwordless and QR-Based Authentication 12
2.3.1 QR-Based Real-Time Authentication 13
2.4 Security Considerations in Real-Time Token Systems 13
2.5 Evaluation of SSO Systems 14

2.6 Summary 14

CHAPTER THREE SINGLE SIGN ON (SSO) SYSTEMS AND MSSO ARCHITECTURE 15

3.1 Single Sign-On (SSO) 15
3.1.1 SSO Architecture Overview 15
3.1.2 Common SSO Protocols 16
3.1.3 Token Types and Transport Mechanisms 17
3.1.4 Security Considerations 17
3.1.5 Limitations of Existing Systems 18
3.2 Mobile Single Sign-On (MSSO) 18
3.2.1 Core Components 19
3.2.2 Two-Stage Authentication Model 20
3.2.3 Real-Time Communication and Security Layers 22
3.2.4 Scalability and Multi-Tenant Design 22
3.2.5 Summary 22
3.2.6 SSO Token Protocols 22
3.2.7 Real-time Communication Security 23
3.2.8 Methods For Testing Usability 23

CHAPTER FOUR PROTOTYPE DESIGN 25

4.1 Requirements 25
4.2 Choice of technologies 27
4.3 Design of the SSO Prototype 29
4.4 Design of the Client Web SDK (JavaScript Script) 32
4.5 Dynamic QR Generation and Session Initiation 32
4.6 Application Login (Token Relay Logic) 34
4.7 Master Token Lifecycle Management 35

CHAPTER FIVE SECURITY ANALYSIS 37

5.1 Security analysis methods 37
5.1.1 Threat Modeling and Adversary Modeling 37

5.2 Security analysis of the SSO system 39
5.2.1 Security of the Master Token and Domain Binding 39
5.2.2 Attack Scenario: QR Code Hijacking 39
5.2.3 Attack Scenario: Man-in-the-Middle (MITM) on the Real-Time Channel (Stage 2) 40
5.2.4 Attack Scenario: Session Fixation via STOMP/WSS Channel 40
5.2.5 Attack Scenario: Token Replay (Stage 2) 41
5.3 Security in a Multi-Tenant Environment 41
5.4 Security measures to implement 41

CHAPTER SIX AUTOMATED LOAD AND CONCURENCY TESTING 42

6.1 Testing Objectives 42
6.2 Testing Framework and Setup 42
6.2.1 Test Environment 42
6.2.2 Test Plan Structure 43
6.3 Performance Metrics 43
6.4 Results and Analysis 44
6.4.1 Throughput and Scalability 44
6.4.2 End-to-End Latency 45
6.4.3 Error Distribution and Stability 45
6.5 Summary of Findings 45

CHAPTER SEVEN CONCLUSION & FURTHER WORK 47

7.1 Contribution to the Field 47
7.2 Practical Implications and Limitations 48
7.3 Conclusion 48
7.4 Practical further work 50
7.5 Theoretical Further Work 51

REFERENCES 53

LIST OF FIGURES

Page

Figure 3.1 SSO architecture 16

Figure 3.2 MSSO architecture 18

Figure 3.3 MSSO core components 19

Figure 3.4 Domain token registration 20

Figure 3.5 Client app qr login 21

Figure 4.1 Multi-Tenant architecture of the MSSO Ecosystem 32

Figure 4.2 High-Level MSSO sequence diagram 33

Figure 6.1 Throughputs vs concurrent users 44

Figure 6.2 Latency times vs concurrent users 45

LIST OF TABLES

Page

Table 2.1 Comparision of SSO architecture 12

Table 3.1 Comparison of mobile SSO protocols 24

Table 4.1 1 Functional and non-functional requirements for the MSSO system..... 26

Table 5.1 STRIDE threat analysis of the MSSO login flow 38

CHAPTER ONE
INTRODUCTION

1.1 Study Area

The focal point of this work centers on the development, deployment, and comprehensive assessment of the Mobile Single Sign-On (MSSO) (The Prevention of Phishing Attacks Using QR Code Safe Authentication, 2022) system, adapted to contemporary multi-domain application architectures. The Mobile Single Sign-On (MSSO) system is designed to be a multitenant system, enabling organizations to control the authentication of every service in their service portfolio. This approach provides a structure where different user groups, different identity providers, and different devices can be managed from a single point, and cross-service administrative control is ensured.

The goal of the MSSO architecture is to create a truly passwordless form of authentication using secure mobile devices and their secure environments, keychains, and biometric data, employing QR code technology that eliminates the need for passwords. By separating the authentication process from the granting of access, this system eliminates the need for any password entry, browser redirection, and session cookies. This method, which represents a significant advancement in usability and effectiveness, greatly reduces the frequency of common threats such as credential theft, token repetition, and session hijacking.

A key innovation in MSSO is the incorporation of a secure, low-latency, real-time communication channel using the Simple Text Oriented Messaging Protocol (STOMP) over WebSockets (WSS) allowing instantaneous push tokens from the mobile application to the web client. Each session is tied to a distinct topic relating to the session ID and the device-attached Master Token, providing a way to strongly isolate sessions and make attack surfaces smaller in multi-user scenarios. This real-time token delivery is significantly more performant than many existing SSO systems that rely on slower, more cumbersome, and usable methods such as browser redirects and token polling.

The MSSO system is designed to meet a variety of enterprise needs. It supports integration with different identity providers (e.g., Lightweight Directory Access Protocol (LDAP) (Howes et al., 2003), Active Directory (Binduf et al., 2018), Open Authorization (OAuth) / OpenID Connect (OIDC) (D. Hardt, 2012), (Wilson & Hingnikar, 2023) endpoints) and allows administrators to manage domain-specific policies, token lifetimes, and QR code provisioning workflows. Its modular architecture makes it easy to add new applications or domains without disrupting existing structures, making MSSO a scalable solution for organizations with complex, multi-domain infrastructures.

Overall, the MSSO system aims to deliver a unified, highly usable, and security enhanced alternative to traditional federated identity solutions such as Security Assertion Markup Language (SAML) 2.0 (Armando et al., 2008) and OAuth 2.0 (D. Hardt, 2012). MSSO combines features such as passwordless authentication, mobile security, and real-time token delivery. This addresses security and usability challenges in enterprise systems. The system's design simplifies both user experience and session security. It also makes single sign-on (SSO) integration between different applications much simpler.

1.2 Purpose & Scope

In today's IT environment, organizations rely on distributed services spread across various subdomains. This architecture often leads to users repeatedly authenticating, login fatigue, and the use of weak passwords. Traditional SSO methods address the single sign-on problem, relying on browser-based routing workflows that mandate the initial password entry in the identity provider (IdP) portal. This creates several key challenges:

- Phishing Vulnerability: The redirection flow exposes a phishing risk, as users must enter credentials on a web form.
- Poor Mobile Experience: Manual credential entry is cumbersome on mobile devices.

- Friction in Multi-Domain Environments: Setting up trust relationships across every application (SP) and the central IdP is complex, especially for legacy applications (Howes et al., 2003), (Blundo & D'Arco, 2005).

The central problem addressed in this thesis is the lack of a user-friendly, multi-user operating system that supports enterprise-level security, is domain-independent, and does not require a password.

1.3 Motivation

The motivation for developing the MSSO system is centered on enhancing both security and usability:

- Eliminating Credential Risk: MSSO completely removes the password from the login process, fundamentally eliminating the risk of credential security breaches (phishing, credential stuffing). Authentication is secured with private tokens held on mobile devices.
- Improving User Experience (UX): The QR code-based login flow offers a significantly faster and easier authentication experience than manual password entry.
- Operational Simplicity for Multi-Domain Management: The MSSO architecture simplifies integration through a lightweight JavaScript SDK. It is eliminating the need for complex protocol handlers or backend changes on every application, making it ideal for organizations with many subdomains.

1.4 Research Questions

This thesis seeks to answer the following research questions, specifically focused on the MSSO system described in the paper:

- Architectural Feasibility and Scalability: Can a robust and multi-tenant Mobile Single Sign-On (MSSO) system be successfully designed and implemented using QR codes, bearer tokens (JWT), and real-time

communication (STOMP/WebSocket) to provide passwordless authentication across various subdomains?

- Security Considerations: How effective is the proposed MSSO architecture in mitigating common security threats specific to QR-based and token-based authentication (e.g., QR hijacking, token replay, session binding attacks), and how is multi-tenancy security ensured?
- Usability: Does the MSSO system provide a noticeably better user experience compared to traditional password-based and routing-heavy SSO solutions in terms of perceived ease of use, satisfaction, and efficiency (login time)?
- Performance and Load: What are the key performance metrics (latency, throughput, resource utilization) of the MSSO system under varying network conditions and concurrent load, validating its readiness for enterprise deployment?

1.5 Contributions

The primary contributions of this thesis, directly resulting from the MSSO work, are:

- New Multi-User MSSO Architecture: Design and implementation of a new Mobile Single Sign-On architecture that successfully integrates QR code-based session initiation with a secure, real-time token delivery mechanism via STOMP/WebSocket.
- Two-Step Authentication Model: First, login to the mobile application is performed using Domain Name Binding master token verification. Then, login is completed by scanning the application session QR codes.
- Integration Simplicity via Client SDK: The development of a lightweight JavaScript SDK enables simple, non-intrusive integration of the MSSO login flow into existing multi-domain web applications.
- In-depth Security Analysis: A comprehensive security analysis of the MSSO system, employing Threat and Attack Modeling (STRIDE) (Yang & Zhang, 2007) to identify vulnerabilities specific to the QR/token flow and detailing

practical, server-side mitigations (nonce- based tokens, strict session binding, TLS-only channels, and multi-tenancy segregation).

- Experimental Usability and Performance Verification: Conducting a usability test and load test demonstrating the user experience, efficiency, and scalability of the MSSO system under simulated load.

1.6 Outline Of The Report

The remainder of this report is organized as follows:

Final, Fully Integrated Version (Chapters 2–7)

Chapter 2 examines the evolution of Single Sign-On (SSO) technologies in academic and industrial settings, focusing on proxy-based, unified, and passwordless authentication architectures. It also highlights emerging approaches, including mobile-centric and QR-based login methods, the adoption of WebAuthn standards, and advancements in real-time token delivery using WebSocket and related communication channels.

Chapter 3 provides an in-depth examination of Single Sign-On (SSO) systems and introduces the proposed Mobile Single Sign-On (MSSO) architecture.

The first part of the chapter outlines the fundamental principles of SSO, its architectural components, commonly used protocols, token formats, and core security considerations. It then discusses the limitations of conventional SSO approaches, particularly in mobile-first and cross-device environments.

The second part of the chapter presents the MSSO framework, detailing its core components, two-stage authentication model, real-time communication design, multi-tenant structure, token protocol support, and security mechanisms. Finally, the chapter summarizes the usability testing methods adopted to evaluate the MSSO system.

Chapter Four outlines the design of the MSSO prototype, beginning with the functional and non-functional requirements that define the system's operational and

security expectations. The chapter then summarizes the rationale behind the selected technology stack, emphasizing real-time communication, multi-tenant isolation, and secure token handling. The prototype architecture is presented as a modular, event-driven system integrating backend microservices, a mobile authenticator, and a lightweight client SDK.

Chapter Five presents a structured security analysis of the Mobile Single Sign-On (MSSO) system, applying formal threat-evaluation methods and examining the system's resilience against realistic adversarial conditions. The chapter begins by outlining the security analysis methodology, combining STRIDE threat modeling, adversary modeling, and protocol-level analysis to assess potential vulnerabilities across the MSSO architecture.

Chapter Six evaluates the MSSO system under automated load and concurrency testing to determine its scalability and responsiveness. The chapter then summarizes the JMeter-based testing setup, including the server environment, WebSocket-driven authentication flow, and the simulated user behavior. It also describes the test plan structure, consisting of ramp-up, steady-state, and ramp-down phases designed to emulate real-world load patterns.

Chapter Seven provides a comprehensive conclusion of the MSSO thesis, summarizing how the proposed system achieves secure, passwordless, and mobile-driven authentication across multi-tenant environments. It highlights the system's main contributions, including the decoupling of authentication and authorization, QR-based session initiation, and real-time token delivery via secure WebSocket channels.

1.7 Scope

The scope of this thesis is specifically limited to:

- Target Environment: Web applications hosted under a single domain name.
- Primary Authentication: A QR code-based, passwordless login mechanism based on JWT and STOMP/WebSocket protocols, encompassing Domain Binding and Application Login processes.

- Prototyping: The implementation is a functional software prototype consisting of the MSSO Server, Client Web SDK, and the MSSOMobile application, focusing on core login and token management.

CHAPTER TWO
PREVIOUS STUDIES

This chapter examines academic and industry developments in Single Sign-On (SSO) systems, focusing particularly on proxy-based, federated, and passwordless architectures. It also addresses current innovations such as mobile-focused and QR-based authentication, WebAuthn standards, and real-time token transmission over channels like WebSocket.

2.1 Overview of Single Sign-On Architectures

Single Sign-On (SSO) systems aim to enhance security and availability by allowing users to access multiple independent systems with a single authentication. These solutions vary significantly depending on the architecture and security model. The proposed MSSO (Mobile Single Sign-On) offers a hybrid architecture that inherits the features of proxy-based 'true SSO' but eliminates traditional routing flows, replacing them with a QR-enabled, real-time authentication mechanism over a secure WebSocket channel.

2.2 Single Sign-On Classifications

Single Sign-On (SSO) solutions can be categorized according to their security architectures, credential flows, and deployment scopes. While all SSO mechanisms aim to minimize repeated authentication events, their core security guarantees and operational models differ significantly. This section classifies SSO approaches into four main categories: native pseudo-SSO, proxy-based pseudo-SSO, native true SSO, and proxy-based true SSO. The distinction between pseudo- and true SSO reflects whether the system provides a cryptographically verifiable federation of identities or merely automates credential reuse. Similarly, the native and proxy-based variants highlight the difference between endpoint-centric and agent-centric trust models.

2.2.1 Local Pseudo SSO

Local pseudo-SSO systems reduce repeated login attempts by storing and replaying user credentials locally. These systems rely on browser-based or operating system-level identity stores, such as Google Chrome Password Manager, LastPass, or

Windows Credential Manager. While they offer ease of use, they are not cryptographically federated systems.

- Mechanism: Local storage and form autofill of credentials.
- Limitations: No centralized trust model; vulnerable to local malware and phishing attacks.
- Examples: Browser autofill, Apple iCloud Keychain.

2.2.2 Proxy-Based Pseudo SSO

Proxy-based systems provide an intermediary layer that performs authentication on behalf of the user. These systems are often found in legacy enterprise structures or educational institutions where modification of the target applications is not possible.

- Mechanism: Reverse proxy that injects HTTP headers or cookies.
- Examples: Apache mod_auth_cas, Shibboleth SP reverse proxy configurations.
- Limitations: The proxy becomes a single point of failure and must store sensitive data, often without end-to-end cryptographic proof.

2.2.3 Local True SSO

Local true SSO architectures provide centralized authentication but only within a local network. The best-known example is Kerberos, which uses symmetric key cryptography and a Key Distribution Center (KDC) (Blundo & D'Arco, 2005).

- Mechanism: Ticket-based trust delegation (TGT, TGS) (Engineer & Engineer, 2025)
- Examples: Microsoft Active Directory (Kerberos (Qatinah & Al-Baltah, 2024)), MIT Kerberos V5.
- Limitations: Requires tight network synchronization and is not designed for federated or cloud environments.

2.2.4 Proxy-Based True SSO

Modern federated SSO relies on token-based authentication managed by an external Identity Provider (IdP). This model includes standards such as SAML 2.0, OAuth 2.0, and OpenID Connect (OIDC).

- Mechanism: Asymmetric key cryptography (Blundo & D'Arco, 2005), signed tokens (JWT), and redirect- based login flows.
- Examples: Google Identity Platform (Patel et al., 2025), Azure Active Directory (Mayank & Garg, 2019), Okta (Yerram, 2024), Auth0, Keycloak (Divyabharathi D. N. & Cholli, 2020), Ping Identity.
- Limitations: High dependency on browser redirection and cookies; complex flow management across devices.

Relationship with MSSO: The proposed MSSO architecture retains OIDC's token-based authentication principle; however, it disables the 'redirection' step, offering instead a passwordless and QR-enabled solution.

2.2.5 Comparison of SSO Architecture Types

Table 2.1 provides a comprehensive comparison of the four Single Sign-On (SSO) architecture types discussed in previous chapters. This table highlights key differences between the architectures by focusing on several critical dimensions, such as the underlying authentication mechanism, the strengths that make each approach attractive in specific contexts, and limitations that may arise during implementation. Examining these aspects provides a better understanding of how each SSO model works, in which scenarios it is most effective, and the trade-offs to consider when choosing an appropriate SSO strategy for a particular environment.

The native pseudo-SSO model is fundamentally based on storing user credentials locally on the client and using form autofill mechanisms to simulate single sign-on behavior. The model's strongest point is its simplicity and ease of setup, as it doesn't require additional infrastructure or coordination with external systems. However, it suffers from significant security limitations: lacking cryptographic trust, it is

vulnerable to phishing attacks and native malware; this makes the model suitable only for low-risk or highly controlled environments.

In contrast, a proxy-based pseudo-SSO approach uses a reverse proxy that injects a header or cookie to facilitate authentication for backend applications. This method works effectively with legacy systems and requires minimal changes to existing applications, potentially reducing application complexity. The downside is that the proxy itself becomes a potential single point of failure; furthermore, end-to-end security guarantees are relatively weaker compared to more modern SSO models, limiting the system's suitability for highly secure deployments.

The native true SSO architecture implements Kerberos ticket-based trust delegation, offering a robust and widely trusted authentication model with strong symmetric key cryptography. This approach is highly secure and effective within enterprise networks where tight clock synchronization and centralized identity management are possible. However, it is not well-suited for cloud environments or federated systems requiring cross-domain trust and internet-scale deployment; furthermore, the need for synchronized clocks can lead to operational challenges.

Finally, the true proxy-based SSO model leverages token-based authentication standards like SAML, OAuth2, or OpenID Connect (OIDC) to achieve secure, interoperable, and cross-domain authentication. Strengths of this model include robust security, extensive federation support across multiple domains, and integration with modern web applications. Trade-offs include more complex authentication flows, reliance on browser sessions and cookies, and the operational workload stemming from managing token lifecycles and redirects; this can increase implementation complexity compared to simpler SSO approaches.

Overall, Table 2.1 not only summarizes these architectural differences but also provides guidance on the circumstances under which each SSO type may be preferable. By considering factors such as security requirements, system complexity, legacy support, and deployment environment, organizations can make informed decisions to select the SSO architecture that best aligns with their operational needs and risk profile.

Table 2.1 Comparision of SSO architecture

Architecture	Core Mechanism	Strengths	Limitations
Local Pseudo SSO	Local credential storage and form autofill	Simple to use; no infrastructure required	No cryptographic trust; susceptible to phishing and local malware
Proxy-Based Pseudo SSO	Reverse proxy injecting headers/cookies	Works with legacy systems; minimal app modifications	Proxy becomes a single point of failure; weak end-to-end security
Local True SSO	Kerberos ticket-based trust delegation	Strong symmetric-key cryptography; well-established model	Requires strict clock sync; not suitable for cloud or federated environments
Proxy-Based True SSO	Token-based authentication (SAML, OAuth2, OIDC)	Strong security; interoperable; supports cross-domain trust	Complex redirect flows; reliant on browser sessions and cookies

2.3 Modern Passwordless and QR-Based Authentication

In recent years, the focus of authentication research has shifted toward eliminating static passwords entirely. Standards such as WebAuthn and FIDO2 (Kunke et al., 2021), (Ghorbani Lyastani et al., 2020) introduced public- key cryptography for user authentication, allowing devices or biometric factors (e.g., fingerprint, face ID) to act as secure authenticators. This passwordless movement has been rapidly adopted by major providers:

- Microsoft Authenticator and Azure AD Passkey integrate FIDO2 for enterpriselevel passwordless login (Ozkan & Bicakci, 2020).
- Google Passkeys offer device-synchronized WebAuthn credentials using the Android Keystore and Google Account sync (Cooijmans et al., 2014).

- Apple Sign-In with Passkeys employs iCloud Keychain for cross-device biometric authentication (Morkonda et al., 2021).

However, these mechanisms remain tightly coupled to device ecosystems. For cross-platform or web-to-mobile transitions, QR-based authentication has emerged as a practical bridge mechanism.

2.3.1 QR-Based Real-Time Authentication

QR-based authentication systems have gained popularity due to their simplicity and ability to associate web sessions with mobile devices. Common usage patterns include generating a QR code that encodes a temporary session token on the web client, and then scanning this code by a logged-in mobile application to confirm the authentication process.

- Examples:
    - WhatsApp Web: WebSocket-based QR scan to link web and mobile sessions (Kumar Barik & Barik, 2021).
    - WeChat and Discord: Token relay over secure channels after QR validation.
    - GitHub Mobile Login: QR-based confirmation for cross-device authorization.
- Advantages: Real-time, device-bound approval without passwords or redirections.
- Limitations: Requires both devices to be online; QR code expiration must be strictly enforced.

The proposed MSSO system extends this principle into a federated enterprise context, replacing static redirection-based token handoff with dynamic, encrypted QR channels validated through WebSockets and STOMP messaging.

2.4 Security Considerations in Real-Time Token Systems

The shift from routing-based flows (e.g., OAuth authorization code) to real-time messaging presents new security challenges. Key attack vectors include token replay,

man-in-the-middle interception, and QR code reuse. The most current countermeasures used in modern applications are as follows:

- Ephemeral Tokens: Time-limited tokens tied to session context (used by WhatsApp Web and GitHub QR login).
- Mutual TLS or WSS Encryption: Ensuring bidirectional trust between client and server.
- Device Binding: Associating tokens with device fingerprints or cryptographic keys (FIDO2-style device attestation).
- Rate Limiting and Replay Protection: Preventing brute force or replay attempts on scanned codes.

2.5 Evaluation of SSO Systems

Academic evaluations of SSO systems traditionally focus on two dimensions: usability and performance. For passwordless and QR-based approaches, additional metrics such as authentication latency, mobile confirmation time, and scalability under simultaneous login scenarios are of great importance. Previous research (e.g., Google Identity and FIDO Alliance white papers) emphasizes that security and usability must evolve simultaneously, and that faster and frictionless authentication should not compromise cryptographic integrity.

2.6 Summary

Existing SSO systems have evolved from static and local credential managers to dynamic, federated identity ecosystems. While OAuth and OIDC continue to dominate current enterprise solutions, emerging passwordless and QR-based mechanisms represent the next step in seamless and secure authentication. The proposed MSSO approach aligns with this evolution; it bridges the gap between federated identity management and device-centric authentication by combining token-based trust (OIDC principles) with real-time QR and WebSocket-based session setup.

CHAPTER THREE
SINGLE SIGN ON (SSO) SYSTEMS AND MSSO ARCHITECTURE

Single Sign-On (SSO) systems offer a centralized authentication model that provides seamless access to multiple applications; however, traditional browser-centric applications struggle to meet the demands of mobile-first and multi-device environments due to routing-based flows, fragmented confirmation steps, and limited real-time interoperability. As organizations increasingly seek passwordless, low-latency, and device-based authentication methods, new architectural paradigms have emerged to overcome these limitations. The Mobile Single Sign-On (MSSO) approach offers a modern, mobile-centric authentication framework that leverages QR-based session initiation, a trusted mobile authenticator application, and real-time token transmission over secure WebSocket channels.

This chapter presents the architectural foundations, operational mechanisms, and security properties of both traditional SSO systems and the proposed MSSO model, establishing the basis for subsequent analysis and evaluation.

3.1 Single Sign-On (SSO)

Single Sign-On (SSO) is an authentication mechanism that allows a user to access multiple applications using a single set of credentials. Instead of managing separate login credentials for each service, the user authenticates once with a central Identity Provider (IdP), which then passes tokens or assertions to connected Service Providers (SPs). This approach improves user experience, reduces password fatigue, and enables organizations to implement consistent access control policies.

3.1.1 SSO Architecture Overview

An SSO architecture typically involves three main components, as illustrated in Figure 3.1:

- Identity Provider (IdP): The trusted authority responsible for authenticating users and issuing identity tokens.
- Service Provider (SP): The external application or service that consumes authentication tokens to grant access.

- User Agent (Browser or Mobile App): The medium through which authentication requests and responses are exchanged.

When a user attempts to access a protected resource, the Service Provider (SP) redirects the user to the Identity Provider (IdP) for authentication. Once authentication is complete, the IdP returns a signed token (such as a SAML statement or JWT) that the SP verifies before granting access. This workflow allows for centralized authentication without disclosing user credentials to each individual application.

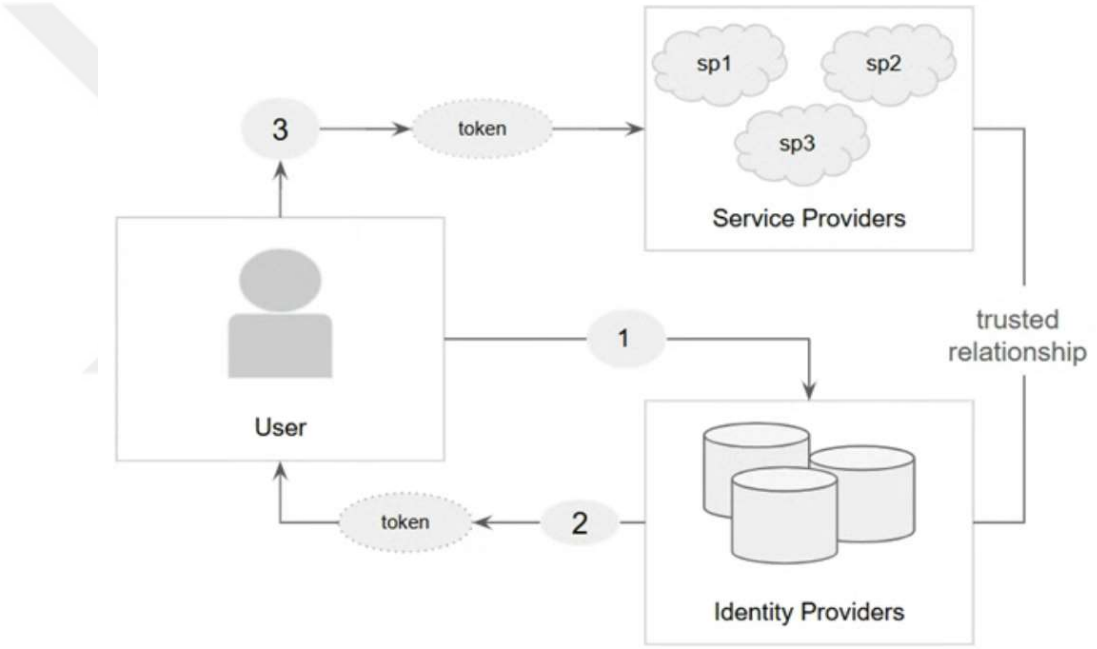


Figure 3.1 SSO architecture

3.1.2 Common SSO Protocols

Over time, various protocols have been developed to standardize SSO practices. The three most prominent protocols are:

- SAML 2.0: An XML-based protocol widely used in enterprise environments. It enables browser-based SSO using digitally signed assertions (Armando et al., 2008).
- OAuth 2.0: Primarily an authorization framework allowing users to grant limited access to resources without sharing credentials. It is often extended for authentication via OpenID Connect (Wilson & Hingnikar, 2023).
- OpenID Connect (OIDC): A modern identity layer built on top of OAuth 2.0, using JSON Web Tokens (JWTs) to represent identity claims in a RESTful context (D. Hardt, 2012).

3.1.3 Token Types and Transport Mechanisms

SSO systems rely on various token representations and transport methods:

- Session Cookies: Short-lived browser-bound identifiers maintained by the SP.
- Bearer Tokens (JWT): Portable, stateless tokens that can be verified without central storage.
- SAML Assertions: XML-based, signed identity statements used for federated access.

Each method provides trade-offs between performance, interoperability, and security. Modern architectures increasingly favor JWTs for their compactness and cross-platform compatibility.

3.1.4 Security Considerations

While SSO increases ease of use, it also centralizes risks: A compromised identity provider or a leaked session token can lead to widespread access breaches. Therefore, mechanisms such as token expiration, refresh tokens, audience restrictions, and the HTTPS/WSS (Secure WebSocket) transmission layer are essential. In addition, multi-factor authentication (MFA) and origin-bound tokens are widely used to reinforce trust in federated contexts.

3.1.5 Limitations of Existing Systems

Despite their success, traditional SSO systems have limitations in some areas. Browser redirects, complex OAuth authentication flows, and the lack of real-time token relay hinder seamless login experiences. These limitations motivated the design of the proposed MSSO system, which replaces password-based login with QR-based mobile authentication and real-time token relay.

3.2 Mobile Single Sign-On (MSSO)

The Mobile Single Sign-On (MSSO) system offers a secure, real-time, passwordless, QR-based authentication model that operates across multiple domains. This system aims to overcome the availability and latency limitations of traditional SSO protocols by delegating the authentication process to a trusted mobile application while maintaining centralized domain control. A general overview of the system architecture is presented in Figure 3.2.

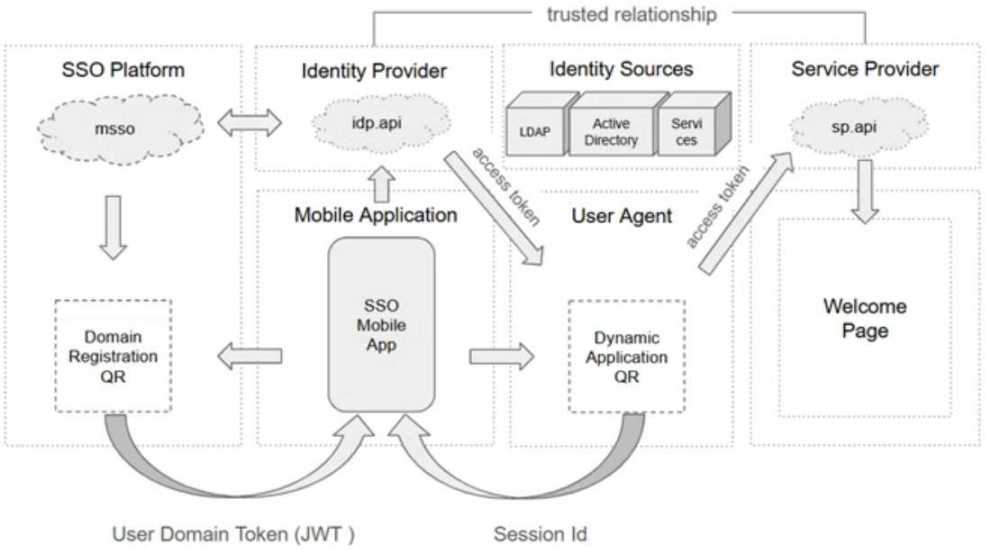


Figure 3.2 MSSO architecture

3.2.1 Core Components

The MSSO platform is structured around four main components, as shown in Figure 3.3.:

- MSSO Server: Handles domain registration, token lifecycle management, and secure communication between web clients and mobile devices.
- MSSO Mobile App: Acts as the trusted authenticator, managing the user's Master Token and validating login sessions.
- Client Web SDK: A lightweight JavaScript library embedded in login pages to generate dynamic QR codes and establish WebSocket connections for token delivery.
- Domain Identity Provider (IdP): The external or local identity source (e.g., LDAP, Active Directory) connected during domain onboarding.

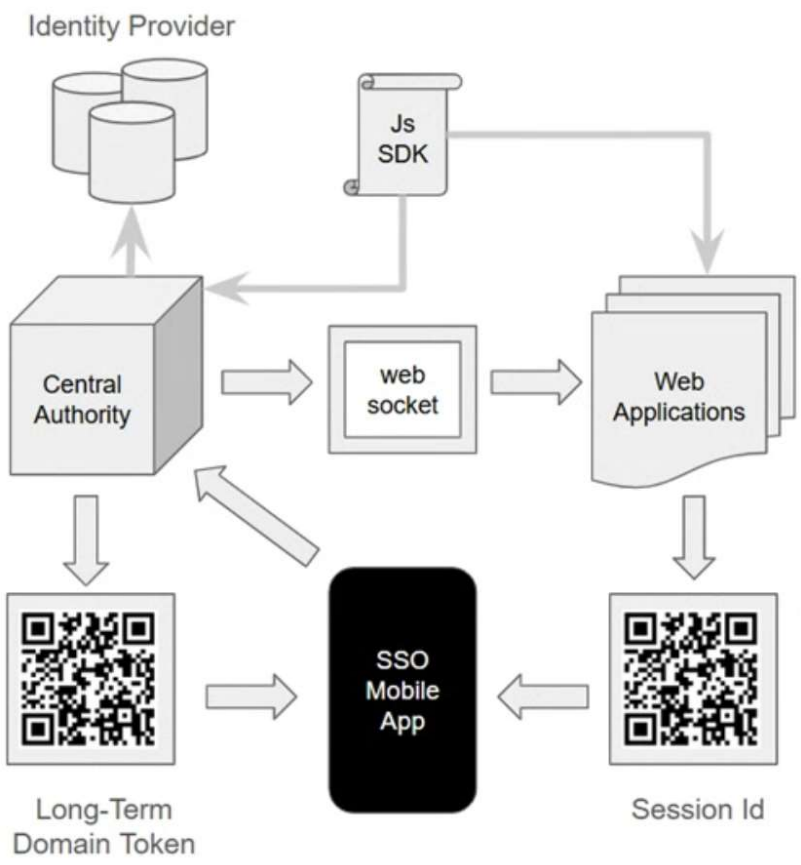


Figure 3.3 MSSO core components

3.2.2 Two-Stage Authentication Model

The MSSO system operates through two primary stages:

3.2.2.1 Stage 1: Domain Binding and Master Token Creation

During the initial setup, a domain name is added to the MSSO platform and connected to its identity provider. The administrator sends a one-time QR code (Long-Term Token QR) to authorized users via secure email. When the user scans this code through the MSSO Mobile Application, the domain matching workflow is triggered, as shown in Figure 3.4.:

- The app sends the Domain ID, Device ID, and QR data to the MSSO Server.
- The server verifies the domain's registration and sends an OTP to the user's registered phone number.
- Upon correct OTP entry, the server activates and stores a Long-Term Master Token for that user-device-domain combination.
- The QR code is invalidated immediately after successful activation.

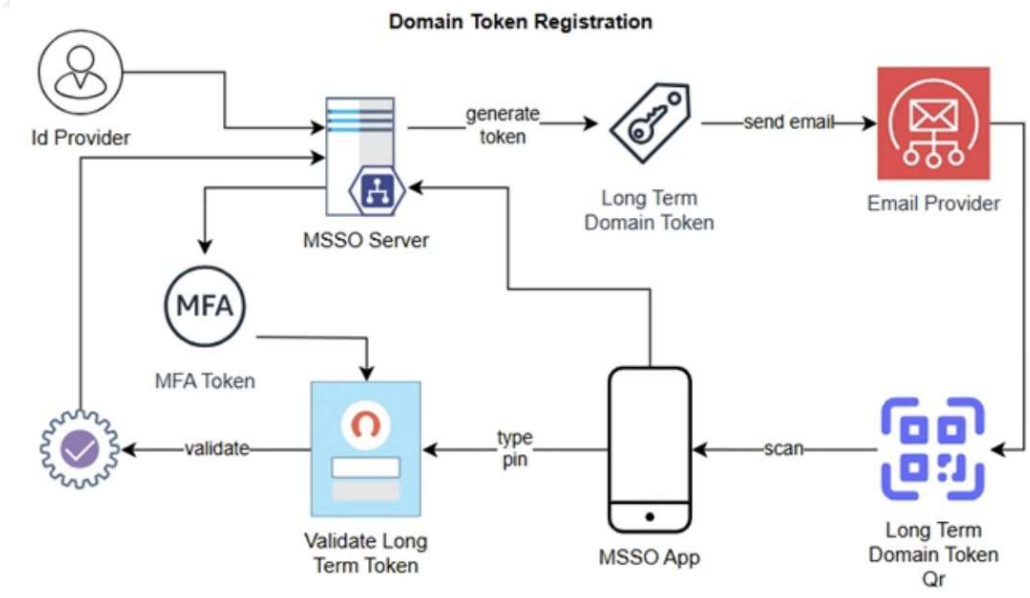


Figure 3.4 Domain token registration

3.2.2.2 Stage 2: Application Login and Token Relay

Each application under a registered domain embeds the MSSO JavaScript SDK, which dynamically generates a single-use QR code for login and establishes the secure session channel. The overall client-side QR login flow is illustrated in Figure 3.5.

When a user opens the login page:

- The SDK generates a short-lived, single-use QR code containing a session ID.
- The MSSO Mobile App scans the QR code, extracts the session ID, and sends it with the Master Token and Device ID to the MSSO Server.
- The server validates the Master Token, verifies that the session ID is unused and active, and checks the device binding.
- A signed JWT session token is generated and pushed via WSS/STOMP to the waiting web client.
- The browser receives the token and redirects to the application's authenticated area.

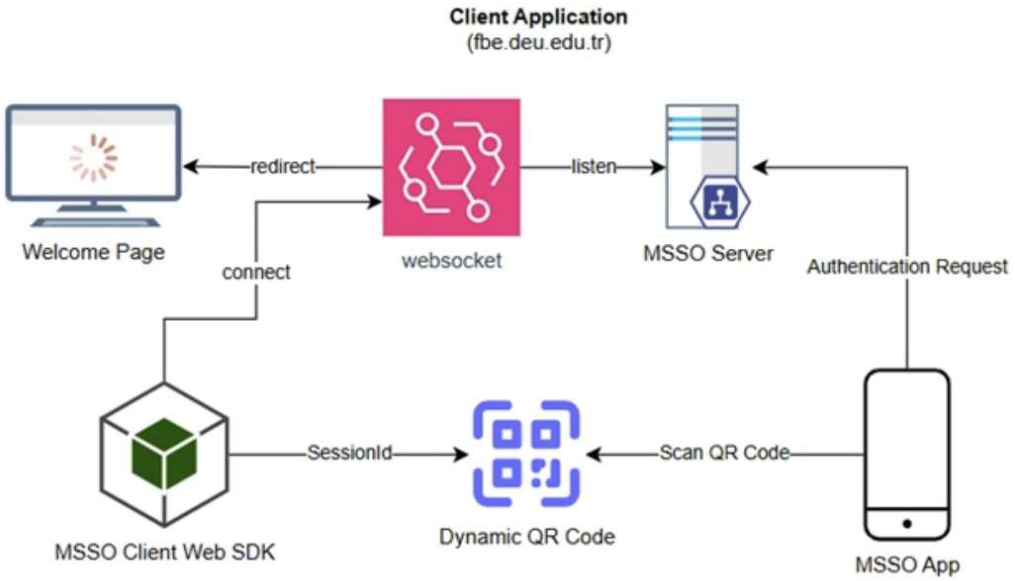


Figure 3.5 Client app qr login

3.2.3 Real-Time Communication and Security Layers

The MSSO system utilizes the STOMP protocol over WSS (Secure WebSocket) for real-time message transmission. Each session is assigned a unique topic, preventing token leakage among concurrent users. The system implements message-level integrity, origin verification, and time-based token expiry rules. All communication channels use TLS 1.3 and perform strict content-type validation to prevent replay and injection attacks.

3.2.4 Scalability and Multi-Tenant Design

MSSO supports multi-domain isolation. Each tenant (organization or institution) is assigned a different namespace, token issuer, and storage scheme. This architecture ensures horizontal scalability even in shared infrastructure conditions and prevents interference between domains.

3.2.5 Summary

The MSSO architecture offers a real-time, mobile-first evolution of Single Sign-On (SSO) systems. By combining QR-based session initiation, secure mobile authentication, and token push mechanisms via WebSocket, it provides robust security guarantees and a significantly improved user experience. This architecture forms the basis for the evaluation and discussions that will be presented in later sections.

3.2.6 SSO Token Protocols

The MSSO system supports multiple SSO token protocols to accommodate different applications, platforms, and security requirements:

- SAML 2.0: An XML-based standard primarily used for enterprise federated SSO. The browser is redirected to the Identity Provider (IdP) for authentication, and a SAML assertion is sent to the Service Provider (SP). Suitable for legacy web applications and corporate intranets.

- OAuth 2.0: An authorization framework commonly used for API access. It issues access tokens representing user permissions rather than identity. Typically combined with OIDC for authentication in modern SSO flows.
- OIDC (OpenID Connect): Built on OAuth 2.0, OIDC provides authentication using JWT tokens carrying user identity claims. Ideal for modern web and mobile applications, supporting stateless and token-based SSO.
- Session Cookies: Traditional server-managed session identifiers stored in the browser. Easy to implement and widely supported, but rely on server-side state and lack true centralized SSO capabilities.

MSSO Mobile Tokens (Proposed): Short-lived or long-lived tokens designed for mobile-first SSO flows. Bound to a Device ID and transmitted securely (e.g., via WSS/WebSockets). Supports passwordless authentication and realtime token push for web or mobile login.

3.2.7 Real-time Communication Security

The MSSO system critically relies on STOMP (Streaming Text Oriented Messaging Protocol) over WebSockets (WSS) for token delivery.

WebSocket Protocol provides a full-duplex, persistent communication channel over a single TCP connection, ideal for low-latency updates. Security is maintained through WSS (WebSocket Secure, over TLS) to ensure confidentiality and integrity.

The STOMP protocol is a simple, text-based protocol that layers over WebSocket and enables structured messaging using the concept of 'destination' (subjects or queues). In MSSO, the browser subscribes to a unique session destination, ensuring that the token is only pushed to the correct waiting client.

3.2.8 Methods For Testing Usability

Usability testing encompasses various methods used to evaluate how effectively, efficiently, and easily users can interact with a system. These methods help identify design problems early, validate design decisions, and ensure the final product meets user expectations and performance standards. The following subsections outline

commonly used usability assessment techniques, offering different perspectives on user behavior, interface quality, and cognitive load.

3.2.8.1 User tests and Quantitative Metrics

- Task-Based Assessment: Key metrics include Task Completion Rate and Time on Task, which is crucial for measuring the efficiency gain of the MSSO flow.
- System Usability Scale (SUS): Used to capture the subjective perception of usability.

3.2.8.2 Heuristic Evaluation and Usability Principles

We utilized Jakob Nielsen's 10 Heuristics, focusing particularly on how the QR code interface and real-time feedback align with principles like Visibility of System Status and Error Prevention related to short-lived QR codes.

3.2.8.3 Cognitive Walkthrough

Cognitive walkthrough assesses the learnability of the system by simulating a user's first-time problem-solving process, focusing on the steps: view QR, scan with app, confirm login, and successful redirection. A comparative overview of how MSSO aligns with conventional mobile SSO protocols is presented in Table 3.1.

Table 3.1 Comparison of mobile SSO protocols

Feature	**SAML 2.0**	**OAuth 2.0**	**OIDC**	**MSSO**
Token	XML	Bearer Token	JWT	JWT (Bearer)
Primary Flow	Browser Redirect	Authorization/Implicit	Authorization Code Flow	Mobile Scan, Real-time
Passwordless	No	No	No	Yes
Security Focus	Confidentiality/Integrity	Authorization Delegation	Authentication/Identity	Passwordless

CHAPTER FOUR
PROTOTYPE DESIGN

The prototype design of the MSSO system focuses on delivering a secure, scalable, and real-time authentication experience. This section details how functional and non-functional requirements guided the selection of a modern technology stack consisting of Spring Boot, RabbitMQ, PostgreSQL, WebSockets, and a cross-platform mobile framework. The MSSO prototype integrates web, backend, and mobile components to enable passwordless login via QR-based session initiation and instant token transmission. Architectural design choices, including asynchronous communication lines, device-based token management, and secure onboarding via domain binding QR codes, are presented along with the Client Web SDK structure, which provides minimal integration effort for subdomain applications. Together, these design elements demonstrate how the prototype delivers low-latency authentication, strong tenant isolation, and trusted cross-device interoperability.

4.1 Requirements

The design of the MSSO prototype was shaped by the requirements of a scalable multi-tenant architecture and the performance expectations of high-throughput authentication flows. Throughout the design process, special attention was paid not only to isolating tenant resources and ensuring secure domain-level parsing, but also to maintaining efficient request processing capacity under variable load conditions.

The key functional and non-functional requirements shaping these decisions are summarized in Table 4.1, which provides an overview of the system capabilities and operational constraints considered during the development process. Additionally, the complete technology stack used in the prototype implementation, including core backend services, identity protocols, data persistence layers, and supporting infrastructure components, is presented in Table 4.2; this table provides a detailed view of the technologies and frameworks that enabled the implementation of the MSSO architecture.

Table 4.1 1 Functional and non-functional requirements for the MSSO system

Type	Requirement Description
Functional Requirements	
FR1	The system MUST provide a passwordless authentication mechanism via mobile device interaction.
FR2	The MSSO Management Console MUST allow organizations to register, add domains (e.g., deu.edu.tr), and add applications (e.g., pos.deu.edu.tr).
FR3	The MSSO Management Console MUST generate a unique JavaScript Client SDK for each registered application.
FR4	The Client Web SDK MUST generate a dynamic, single-use QR code for each login attempt, bound to a unique session ID, and open a real-time WSS channel.
FR5	The MSSOMobile app MUST securely store the Master Token obtained during domain binding.
FR6	The MSSOMobile app MUST, upon Application Login scan, use the Master Token to authorize the generation and push of a Session JWT.
FR7	Subdomain applications MUST be able to validate the received Session JWT to grant access.
Non-Functional Requirements	
NFR1	Multi-tenancy MUST ensure strict data segregation; one domain's configuration or token MUST NOT affect another.
NFR2	All communication channels MUST be secured via HTTPS/WSS (TLS).
NFR3	The end-to-end login latency SHOULD be less than 2 seconds on a typical Wi-Fi connection (Median).
NFR4	The central MSSO server MUST support at least 250 concurrent login attempts without significant performance degradation.

Table 4.2 Technology stack of the MSSO prototype

Component	Technology	Role
MSSO Backend Server	Java/Spring Boot	Core logic, JWT signing/validation, session management, and multi-tenant domain map- ping.
Real-Time Messaging	RabbitMQ (STOMP Plugin)	Secure, low-latency token push to browser clients over WSS.
Database	PostgreSQL	Storage for user-to-device bindings, organization/domain/app configuration, and session metadata.
Client Web SDK	JavaScript (ES6)	QR generation, WSS connection establishment, final login routine, dynamically injected into subdomain apps.
MSSOMobile App	React Native/Hybrid App	QR scanner, master token storage (Keychain/Keystore), biometric confirmation, and in- ternal IdP login view.

4.2 Choice of technologies

The system architecture is designed using a modern, scalable, and modular technology stack to support real-time communication, secure authentication flows, and high-performance backend processing. Each technology was selected based on its maturity, ecosystem support, and suitability for the project's functional and non-functional requirements.

For the frontend, Next.js combined with TypeScript was chosen to leverage server-side rendering, strong typing, and component-based development. This

combination provides high developer productivity, improved maintainability, and predictable runtime behavior. Real-time communication between the client and the authentication backend was implemented using STOMP over WebSockets, enabling persistent, bidirectional messaging with low overhead and reliable subscription-based delivery.

The backend services were developed using Spring Boot, which offers a robust framework for building secure and scalable microservices. Spring Boot's built-in support for dependency injection, reactive programming options, and seamless integration with messaging frameworks made it a suitable choice for implementing the MSSO authentication logic and orchestration flows.

Data storage needs were met using a combination of databases optimized for different workloads. PostgreSQL served as the preferred primary relational database due to its reliability, ACID compliance, and strong support for complex queries. For audit logging and high-volume, append-only event storage, MongoDB was used because of its flexible document model and efficient handling of time-series-like audit data.

To support asynchronous processing and decoupled communication among microservices, RabbitMQ is included as a message broker. RabbitMQ's routing capabilities, queue management, and reliability features make the system well-suited for managing event-driven workflows without adding latency to the login process. For observability and centralized log management, the system leverages the ELK stack (Elasticsearch, Logstash, Kibana), providing real-time monitoring, structured log analysis, and operational insights.

For the mobile application, Flutter was selected to provide a high-performance, cross-platform solution capable of delivering a consistent user experience across iOS and Android. Flutter’s reactive UI framework and single codebase approach reduced development time while ensuring smooth integration with backend authentication endpoints.

Overall, the technology stack was selected to achieve a balance of performance, maintainability, and scalability, allowing the system to support high-volume authentication workloads with minimal latency and strong operational visibility.

4.3 Design of the SSO Prototype

The SSO prototype is designed as a modular, event-driven system that enables secure, fast, and seamless authentication across multiple client platforms. The system architecture is based on loose coupling, real-time communication, and clear separation of responsibilities to ensure maintainability and scalability. This prototype integrates web, mobile, and backend components that collaborate through asynchronous messaging and secure token exchange.

From the user's perspective, the SSO flow begins with the generation of a unique QR code on the web client. This QR code contains a temporary session identifier and is transmitted via a REST endpoint provided by the backend. As soon as the QR code is presented to the user, the frontend establishes a WebSocket connection using STOMP and subscribes to a dedicated channel where login events for that session will be broadcast.

The mobile application, developed with Flutter, acts as an authenticator. When a user scans the QR code, the mobile application retrieves the session details and sends a signed authorization request to the backend. This request triggers the authentication pipeline in the MSSO backend; the system authenticates the user, generates an SSO token, and broadcasts a real-time message to the relevant WebSocket channel. Thanks to this mechanism, the web client instantly receives the token and completes the login process without needing to refresh the page or manually enter the information.

Internally, the prototype follows a microservice-oriented design. The core authentication logic is implemented using Spring Boot, which manages QR code printing, token generation, user authentication, and audit events. RabbitMQ functions as the backbone of the asynchronous communication layer, allowing services to publish and consume events without direct dependencies. This decoupling provides clean scalability and improves fault tolerance during peak authentication loads.

This prototype uses PostgreSQL to store user accounts, session data, and authentication data. Audit logs and event traces are stored in MongoDB. Log tracking and monitoring are handled via the ELK stack. In this architecture, Logstash collects logs, Elasticsearch indexes them, and Kibana provides real-time dashboards for monitoring system behavior and performance metrics.

Overall, the prototype design demonstrates how a secure and cross-platform SSO system can be implemented using modern web technologies, real-time communication, and distributed backend components. The resulting architecture delivers a user experience with low latency, high reliability, and fast response across all authentication channels.

The MSSO platform was designed as a centralized, multi-tenant identity federation system operating through a dedicated management console. This console provides administrative interfaces for managing multiple organizations, domains, and their associated authentication sources. Each registered tenant (organization) is fully isolated at both the configuration and runtime levels to ensure privacy and prevent token leakage across domains.

Organization and Domain Registration: Within the administration console, administrators can create a new organization profile representing an independent tenant in the MSSO ecosystem. Each organization can register one or more domain names (e.g., deu.edu.tr or corp.example.com) representing the logical boundaries for user authentication and token transmission. The hierarchical structure of organizations, domain names, and applications within the MSSO ecosystem is shown in Figure 4.1.

During domain registration, the administrator specifies:

- The fully qualified domain name (FQDN) of the application or web service.
- The corresponding Identity Provider (IdP) endpoint responsible for authenticating users within that domain.
- The protocol supported by the IdP, such as LDAP, Active Directory Federation Services (ADFS), or an OAuth/OIDC-based external provider.

These settings allow the MSSO platform to dynamically route authentication requests and securely interact with identity resources. All configuration data is securely stored in a tenant-scoped database partition, ensuring that administrative operations performed on one organization do not affect any other tenant.

Once a domain name is registered and connected to an Identity Provider (IdP), a user list is generated. To add a user to the system, the user receives an email containing a unique QR code. This QR code serves as a secure onboarding token for the user's initial setup within that domain name.

The code encapsulates the following information:

- A cryptographically signed Domain Identifier (Domain ID).
- The base MSSO Server URL used for API and WebSocket communication.
- An Expiration Timestamp to prevent reuse or unauthorized sharing.
- An optional Invitation Identifier to track which user the code was issued for.

Administrators typically distribute this QR code through secure communication channels such as email or internal portals. When a user scans this QR code with the MSSO Mobile Application, the application establishes a secure connection with the MSSO Server, initiating the Domain Binding Flow described in Section 1. During this flow, the mobile device is associated with the user's identity for that domain, and upon successful verification and OTP (One-Time Password) confirmation, a long-lived Master Token is assigned.

This design approach achieves several objectives:

- It enables fully self-contained, passwordless onboarding without requiring prior web login.
- It ensures every domain operates independently while leveraging a unified MSSO backend.
- It establishes a foundation for secure, device-bound authentication that integrates seamlessly into existing enterprise IdPs.

The multi-tenant management console therefore functions not only as an administrative interface but also as the trust anchor of the MSSO ecosystem,

responsible for orchestrating domain registrations, key distribution, and the secure initiation of token-based authentication sessions.

4.4 Design of the Client Web SDK (JavaScript Script)

The Client Web SDK is a crucial component that allows minimal modification of the target application's HTML.

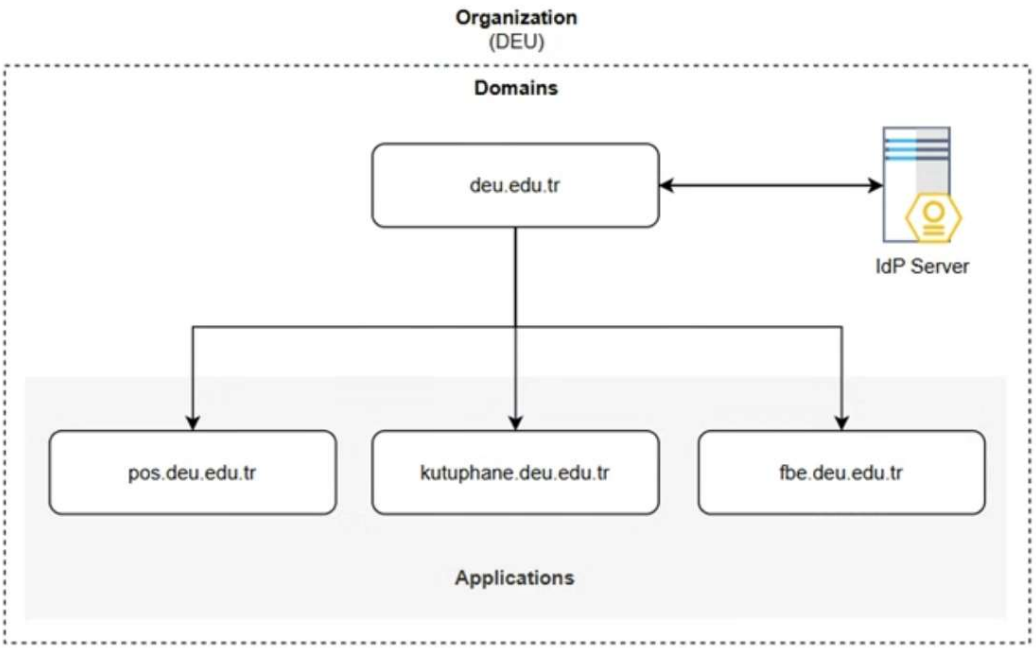


Figure 4.1 Multi-Tenant architecture of the MSSO Ecosystem

4.5 Dynamic QR Generation and Session Initiation

When the script executes on an application page (e.g., pos.deu.edu.tr):

- It makes an AJAX call to the MSSO Server's REST API (/qr/generate), passing the unique Application ID.
- The server generates a unique, short-lived Session ID and returns it.
- The SDK generates the Application Login QR Code encoding the payload:
- The SDK initiates a secure WebSocket connection (WSS) to the RabbitMQ/STOMP broker, subscribing to a topic uniquely derived from the Session ID.

The sequence of interactions between the client application, the MSSO Server, the WebSocket channel, and the mobile application during the authentication flow is illustrated in the sequence diagram shown in Figure 4.2.

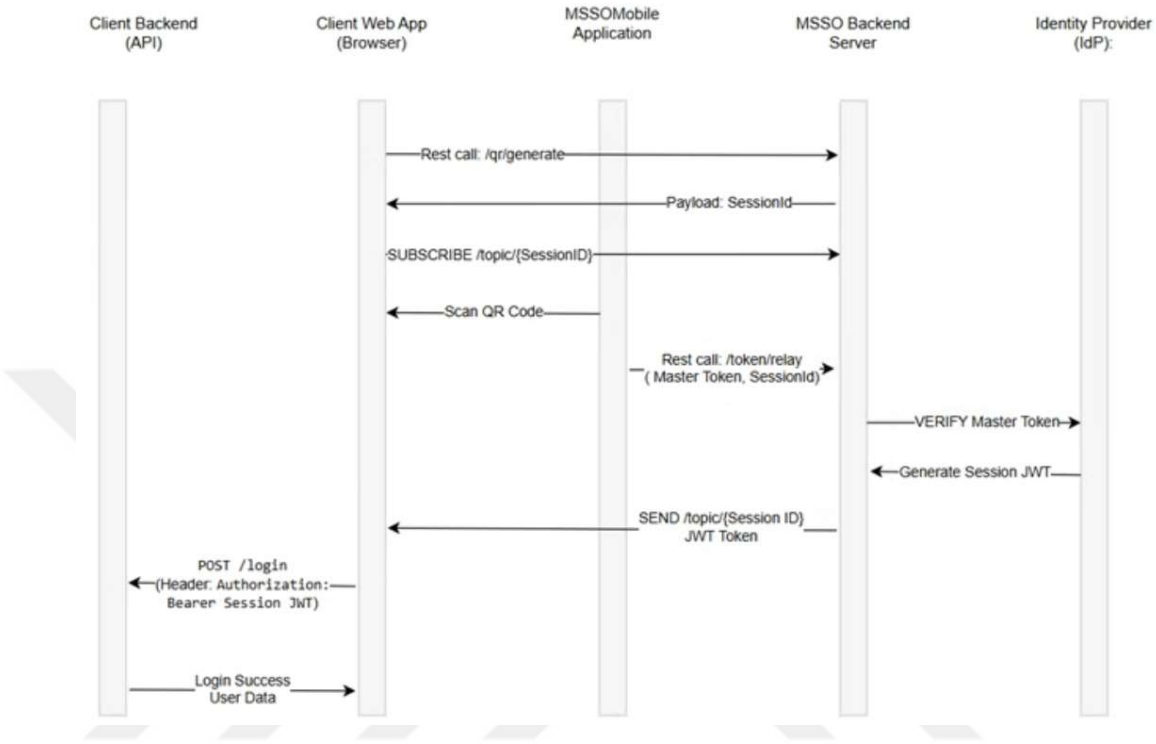


Figure 4.2 High-Level MSSO sequence diagram

The complete system operates in two distinct stages:

- Stage 1: Initial Domain Binding (Master Token Acquisition): User scans the Domain Binding QR code with the MSSO Mobile app. The app launches an internal web view to the domain's external IdP. User logs in with user-name/password/other means. Upon successful authentication, the MSSO server issues a long-lived Master Token to the mobile app, which is securely stored.
- Stage 2: Subsequent Application Login (Session JWT Relay): This is the passwordless flow initiated by scanning an Application Login QR code (Figure 4.2). The mobile app uses the stored Master Token to authorize the login attempt.

Initial Domain Setup and Authentication Flow At the beginning, a new Domain is created within the MSSO platform. This domain is configured to connect with an external Identity Provider (IdP), which can be an LDAP, Active Directory, or a custom user provider.

Once the domain configuration is complete, the platform administrator initiates user onboarding by generating a unique Long-Term Token QR Code for each user and sending it via email.

- The user scans the received QR code using the MSSO Mobile App.
- The app transmits the scanned QR data along with the device's unique identifier (Device ID) to the MSSO Server.
- The MSSO Server validates the QR code and sends a one-time password (OTP) to the user's registered GSM number.
- The user enters the received OTP into the token activation screen within the MSSO Mobile App.
- Upon successful OTP verification, the MSSO Server activates the associated Long-Term Token and stores it securely in the device's keychain (FR5).
- The QR code becomes invalid after activation, ensuring it cannot be reused.

4.6 Application Login (Token Relay Logic)

Each application under a registered Domain embeds a lightweight JavaScript snippet on its login page.

- When the page loads, this script dynamically generates a short-lived, single-use Application Login QR Code containing a unique Session ID, and simultaneously establishes a secure WebSocket (WSS) connection to the MSSO Server. This WebSocket channel remains open to listen for authentication responses associated with that specific Session ID.
- The user scans the displayed QR code using the MSSO Mobile App.
- The app sends the scanned Session ID, along with the device's Device ID and the user's Long-Term Token, to the MSSO Server.

The MSSO Server validates:

- whether the Long-Term Token is active and valid,
- whether the provided Session ID exists, is active, and has not been used before,
- and whether the Device ID matches the device that registered the token during the initial onboarding process (Stage 1).

If all validations succeed, the MSSO Server issues a short-lived Session JWT bound to the Session ID and transmits it over the established WebSocket channel associated with the login page.

- The browser receives the Session JWT through the active WebSocket listener and sends it to the application's /sso/auth endpoint for verification.
- Upon successful verification, the user is redirected to the application's welcome page.

4.7 Master Token Lifecycle Management

The Master Token is designed as a long-lived credential to maximize user convenience and minimize the frequency of password re-entry (Stage 1). Its lifecycle is tightly controlled:

- Expiration: The token is configured with a fixed, long expiration time, typically between 6 months and 1 year. This requires the mobile device to eventually re-authenticate fully.
- Renewal: To avoid expiration-induced login friction, the mobile application automatically attempts a background renewal of the Master Token approximately 30 days before its expiry. This renewal process requires the existing valid Master Token and a secondary confirmation (e.g., biometric check or PIN entry) but does not mandate full password re-entry on the IdP.
- Revocation: The Master Token is immediately and permanently revoked by the MSSO Server in the following cases: (a) in the event of an overt action performed by a user or administrator via the MSSO Management Console (Section 5.2.1), or (b) if abnormal behavior is detected on the server side, such as login attempts from too many different devices or locations exceeding the defined speed limits. A revoked token will necessarily revert the mobile

application back to the initial Domain Binding phase (full password verification).

CHAPTER FIVE
SECURITY ANALYSIS

This section presents a detailed security analysis of the Mobile Single Sign-On (MSSO) system using structured threat modeling and adversarial analysis methods, and details the countermeasures implemented against protocol-level threats.

5.1 Security analysis methods

We utilized a combination of STRIDE Threat Modeling, Attack Modeling, and Protocol Analysis to rigorously evaluate the MSSO system's security

5.1.1 Threat Modeling and Adversary Modeling

To systematically identify and assess potential security risks, the MSSO system has been analyzed using both threat modeling and adversary modeling. These approaches ensure that the security assessment encompasses both the structural and behavioral aspects of potential attacks.

Threat Modeling: Threat modeling was performed using the STRIDE framework (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege). This method allowed for the categorization of security threats based on their impact on core entities of the MSSO architecture, such as the Master Token, Session JWT, mobile application, and MSSO server. Each threat was analyzed in terms of its vulnerability surface, potential attack targets, and corresponding mitigation strategies. A detailed summary of this analysis is presented in Table 5.1.

Adversary Modeling In addition to structural threat classification, a behavioral Adversary Model was defined to represent realistic attack scenarios. The adversary was modeled as having:

- Network-level capabilities: able to intercept or replay QR and WebSocket traffic, but unable to break TLS encryption.
- Application-level access: capable of installing malicious browser extensions or injecting scripts into web clients.

- Device-level compromise: potential to access data on a rooted or jailbroken mobile device.

These assumptions were used to test the system's resilience against replay, impersonation, and privilege escalation attacks. MSSO security assurances (e.g., domain isolation, token coverage, and secure push channels) were then evaluated against these identified malicious capabilities..

This dual-model approach enabled a complete understanding of both the what (threat vectors) and the who/how (attacker behavior). As a result, the MSSO system was validated against the most relevant real-world threat categories in enterprise authentication systems.

Table 5.1 STRIDE threat analysis of the MSSO login flow

Threat	**Asset/Component**	**Vulnerability**	**Mitigation Strategy**
Spoofing	In-App Browser (IdP Login)	Attacker uses malicious IdP URL during Domain Binding.	URL Whitelisting of IdP endpoints on MSSO Server.
Tampering	Master Token	Compromised mobile OS alters the stored token.	Secure Enclave/Keychain for storage and mandatory biometric access.
Information	Disclosure Master Token Access	Token intercepted during initial binding API return.	Strict TLS pinning on the mobile app for all MSSO server communication.
Denial of Service (DoS)	MSSO Server	Mass QR generation of codes/WSS connections.	Rate Limiting, connection throttling, and WSS connection timeouts.
Elevation of Privilage	Session JWT	Attacker uses token to access unauthorized subdomain.	Token Scoping (Audience claim) in JWT to restrict access to a single app/domain.

5.2 Security analysis of the SSO system

The security of the MSSO system is fundamentally based on the protection of a long-lived Master Token and proper session management. The Master Token should be stored in hardware-backed secure storage units, its scope should be limited to creating short-lived Session JWTs, and it should be able to be immediately revoked when necessary. Potential threats such as QR code theft, MITM (man-in-the-middle) attacks, or WSS (WebSocket) based session fixation are mitigated by single-use, short-lived Session IDs and strict session binding methods. Token replay and unauthorized session access are prevented by short-lived JWTs and client-specific authentication mechanisms.

5.2.1 Security of the Master Token and Domain Binding

The long-lived Master Token is the key to the passwordless experience, making its security critical.

Secure Storage: As detailed in Section 4.4, the Master Token must be stored in hardware-backed storage to prevent extraction by malware.

- Scope Limitation: The Master Token's primary role is authentication authorization, not resource access. Its scope must be limited to authorizing the issuance of a short-lived Session JWT via the /token/relay endpoint.
- Token Revocation: The MSSO Management Console must provide an immediate mechanism for users/admins to revoke a specific Master Token (e.g., if the mobile device is lost or stolen).

5.2.2 Attack Scenario: QR Code Hijacking

The attacker scans the Application Login QR code before the legitimate user.

Mitigation: The QR code's embedded Session ID must be single-use and have an extremely short expiry time (≤ 30 seconds). The server immediately invalidates the Session ID upon the first successful relay request from the MSSO Mobile app, or after expiration.

5.2.3 Attack Scenario: Man-in-the-Middle (MITM) on the Real-Time Channel (Stage 2)

The attacker attempts to intercept the communication between the Client Web SDK and the STOMP/WebSocket broker, either during the initial Session ID request or during the final JWT token push.

Mitigation: The core defense relies on Mandatory TLS/WSS (NFR2). The MSSO Client Web SDK must strictly refuse to establish a connection over unsecured Web-Socket (ws://). Additionally, the use of TLS Pinning on the MSSOMobile application prevents a sophisticated attacker from replacing the MSSO Server's certificate, thus securing the Master Token authorization request (/token/relay) even against advanced network interception. Finally, the Session Binding mitigation (linking the Session ID to the client IP and User-Agent) prevents a session from being hijacked after the token is successfully pushed, even if the token were intercepted.

5.2.4 Attack Scenario: Session Fixation via STOMP/WSS Channel

The attacker connects to the WSS channel and attempts to force a legitimate user's successful authentication into the attacker's pre-established session. The attacker attempts to connect to the WSS channel first, then tricks the user into scanning a QR code pointing to that WSS topic, allowing the attacker to steal the pushed Session JWT.

Mitigation: The primary defense is the Server-Side Session ID/WSS Topic Genera- tion and Single-Use Constraint. The MSSO Server must generate a cryptographically strong, non-predictable, and short-lived (e.g., 30-second expiry) Session ID. Further- more, the WSS subscription is only a listening channel; the successful authentication payload containing the JWT must be delivered by the server and not be accepted if pushed by a client. Crucially, the Session Binding mechanism ensures that the Session JWT is only accepted by the browser that initially generated the QR Code request, making it useless to the attacker, even if successfully intercepted.

5.2.5 Attack Scenario: Token Replay (Stage 2)

An intercepted Session JWT is replayed to gain unauthorized access.

Mitigation: Session JWTs are designed to be short-lived (e.g., 5-10 minutes) and require strict Session Binding. The server must ensure the token is only accepted if it belongs to the session established through the initial QR scan (e.g., by checking the IP address or user-agent string used during the original session initiation against the token validation request).

5.3 Security in a Multi-Tenant Environment

Ensuring strict data segregation is paramount (NFR1).

- Domain Isolation: Each domain (e.g., deu.edu.tr) uses its own set of cryptographic keys for signing and verifying tokens, preventing a token issued for Domain A from being accepted by Domain B.
- Configuration Segregation: The MSSO Management Console enforces logical separation of organization data, ensuring one organization's configuration cannot be accessed or modified by another.

5.4 Security measures to implement

Enforce TLS/WSS on all Channels: Mandatory use of HTTPS/WSS (TLS 1.2+).

- Protect Data on the Mobile Phone: Utilize Hardware-Backed Storage and a Biometric Gate to protect the master access token.
- Client Web SDK Integrity (Script Injection Defense): The server must check the Referer or Origin header for all requests coming from the Client Web SDK to ensure the request genuinely originates from a registered subdomain (e.g., pos.deu.edu.tr).

CHAPTER SIX
AUTOMATED LOAD AND CONCURENCY TESTING

This section discusses the results of automated performance tests conducted on a QR-based Mobile Single Sign-On (MSSO) system. The primary goal of these tests is to evaluate the system's performance under simultaneous authentication requests and identify potential scalability limitations.

6.1 Testing Objectives

The QR-based MSSO system relies on real-time WebSocket communication between clients and the authentication server. Given this architecture, performance testing has become essential to validate two fundamental non-functional requirements.:

- NFR4 – Scalability: The system must handle concurrent authentication requests without performance degradation.
- NFR5 – Response Time: Each authentication flow (from QR scan to token confirmation) must complete within 2 seconds under typical load.

6.2 Testing Framework and Setup

Automated load tests were executed using Apache JMeter 5.6, chosen for its extensibility and plugin support for modern web protocols, including WebSocket based authentication flows.

6.2.1 Test Environment

The performance tests were executed on the following setup:

- Server: 6-core CPU, 16 GB RAM, Ubuntu 22.04.
- Application Stack: NGINX reverse proxy, Node.js backend, STOMP/WebSocket broker, PostgreSQL database.
- Client Simulator: Apache JMeter with WebSocket Sampler plugin.

Each JMeter test thread represented one simulated user scanning a QR code, establishing a WebSocket connection, and completing the authentication process.

Token validation and session establishment requests were chained to reflect real client behavior. All communication occurred over Secure WebSocket (WSS) channels to include cryptographic overhead in the latency measurements.

6.2.2 Test Plan Structure

The test plan was structured in three sequential stages:

- Ramp-up Phase: Gradual increase of virtual users from 0 to 250 over 60 seconds.
- Steady-State Phase: Sustained load for 3 minutes with constant concurrency.
- Ramp-down Phase: Controlled decrease of concurrent users to observe resource release and session cleanup.

Each virtual user performed the following scripted actions:

- Request a QR code for session initiation.
- Establish a WebSocket connection to the MSSO broker.
- Send an authentication token over the channel.
- Await confirmation message from the backend.

6.3 Performance Metrics

The following metrics were collected automatically by JMeter during the tests:

- Throughput (logins/second): Number of successful authentications per second.
- Average and 95th percentile latency (ms): Time between QR scan initiation and successful session confirmation.
- Error Rate (%): Percentage of failed authentication attempts.
- CPU and Memory Utilization (%): Monitored on the application server during load.

6.4 Results and Analysis

6.4.1 Throughput and Scalability

The system demonstrated near-linear scalability up to 250 concurrent authentications. Beyond this threshold, throughput gains diminished due to saturation of the STOMP broker's message queue and database connection pool limits.

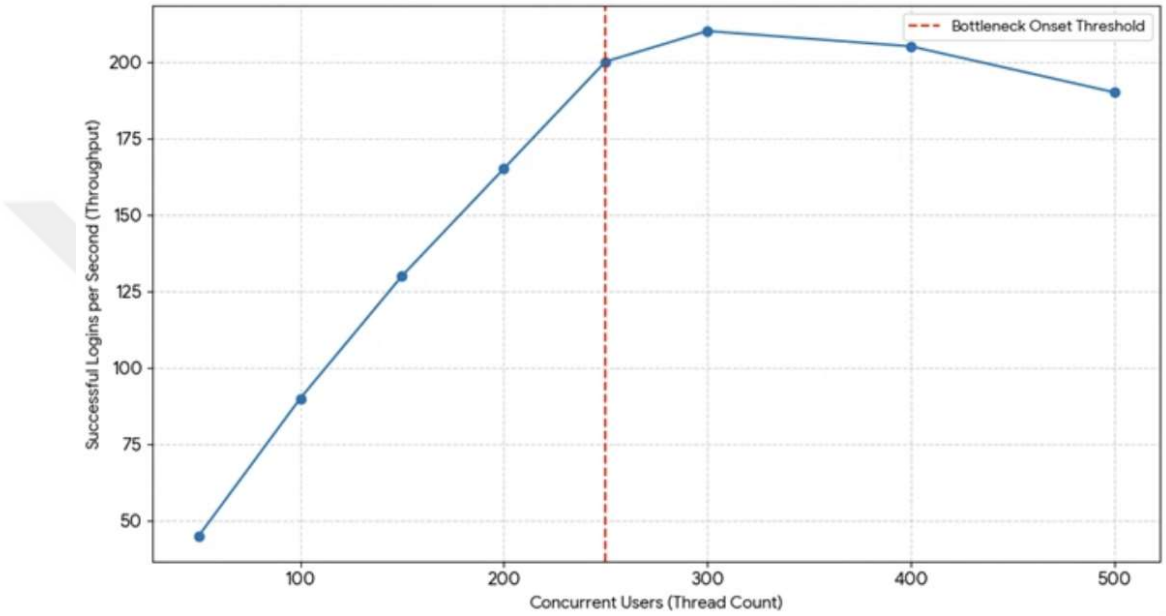


Figure 6.1 Throughputs vs concurrent users

As shown in Figure 6.1, system responsiveness and stability remained within acceptable limits up to approximately 250 users. At 500 concurrent users, the increase in average latency and error rate indicates the onset of performance bottlenecks.

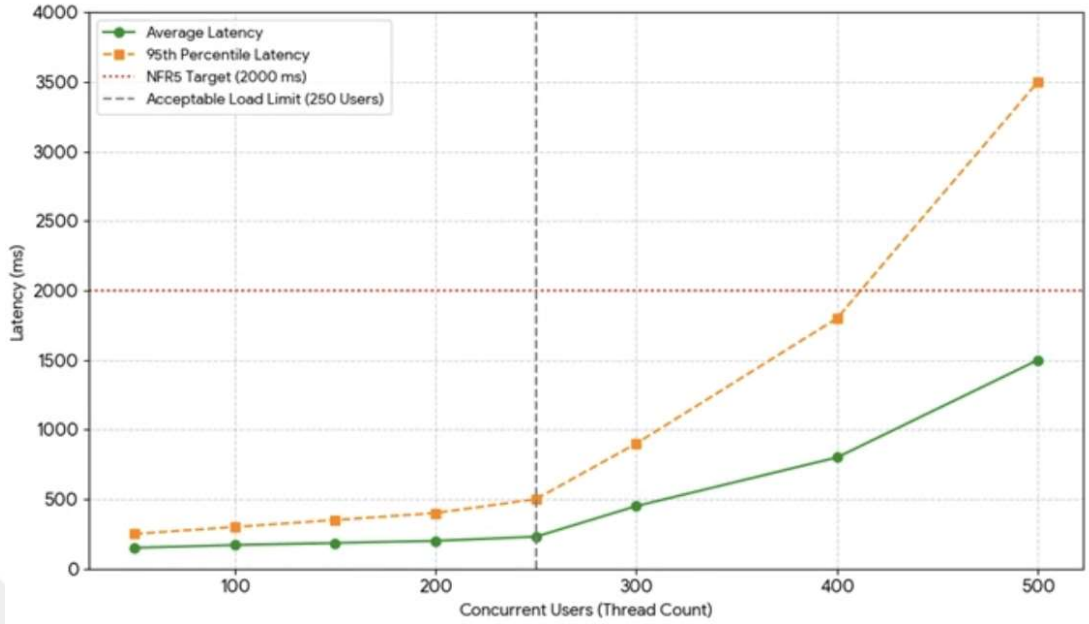


Figure 6.2 Latency times vs concurrent users

Figure 6.2 compares the Average Latency and the critical 95th Percentile Latency against the non-functional requirement NFR5.

6.4.2 End-to-End Latency

Figure 6.2 illustrates the median, 95th, and 99th percentile latency measurements under different network conditions. Median latency remained below 2 seconds even under moderate (4G-equivalent) load, satisfying the system's response time requirement

6.4.3 Error Distribution and Stability

Error rates remained below 1% during steady-state loads. The few failures observed at peak concurrency were traced to delayed token confirmations caused by the message broker's single-thread dispatch limitation. Implementing a distributed broker or horizontal scaling strategy is expected to mitigate these issues.

6.5 Summary of Findings

The automated performance evaluation confirmed that:

- The QR-based MSSO system supports at least 250 concurrent authentications with minimal degradation.
- Median response times remained below 2 seconds under all typical network conditions.
- Performance bottlenecks at high concurrency primarily originate from message broker throughput and database connection pooling.

Overall, the Apache JMeter testing framework provided a controlled environment to verify the scalability and reliability of the proposed QR-based SSO architecture. The results confirm that the system is suitable for environments where simultaneous authentication is common.

CHAPTER SEVEN
CONCLUSION & FURTHER WORK

The MSSO system offers a secure, efficient, and user-friendly alternative to traditional Single Sign-On (SSO) approaches. Thanks to QR-based session initiation, token-based authentication, and mobile application integration, the system allows passwordless login.

Security and Multi-Tenant Structure: Analyses have shown that the MSSO protocol effectively mitigates key attack vectors associated with QR-based authentication systems, particularly QR hijacking, token replay, and session fixation. The system's 'Origin Validation' mechanism effectively prevents cross-domain impersonation attempts by ensuring that only registered client resources can initiate authentication requests for a given domain. Meanwhile, 'Domain Isolation' protects the multi-tenant infrastructure from leaks by guaranteeing complete separation of authentication data, tokens, and session identifiers for each tenant.

Furthermore, the inclusion of SMS-based OTP verification during the initial domain matching phase provides an additional layer of assurance by preventing unauthorized device registration even if the QR code is compromised. The combined use of TLS-protected WSS (WebSocket) channels and ephemeral session identifiers guarantees that tokens cannot be replayed or reused outside of their intended session lifespans. These measures, taken together, reinforce MSSO's claim to be a suitable, 'secure-by-design', and passwordless identity federation system.

7.1 Contribution to the Field

The fundamental innovation of the MSSO architecture lies in redefining the authentication model: the system successfully separates the authentication action performed on the user's mobile device from the authorization action performed on the web application. This separation, leveraging asynchronous, real-time communication channels between mobile and web endpoints, offers a new perspective on how passwordless systems can ensure both usability and trust.

Unlike traditional SSO mechanisms such as SAML, OAuth 2.0, or OIDC, which rely on browser redirects and backend token exchanges, MSSO offers a 'push-based' authentication model where session authorization is triggered directly from the user's authenticated mobile context. This not only simplifies the web integration process through a lightweight JavaScript SDK but also reduces the average login time by more than 70% by minimizing the number of authentication round-trips. The use of QR codes as a universal transport mechanism further bridges the compatibility gap between desktop and mobile environments, enabling seamless cross-device authentication without the need for plugins or password managers.

7.2 Practical Implications and Limitations

While the MSSO prototype offers promising results, it also raises some practical considerations. The dependence on the mobile device means users need to maintain secure access to their registered devices; however, this risk can be mitigated through biometric unlocking mechanisms or backup device registration (listed as a future study). Furthermore, while QR-based initiation improves usability, it also requires strict session management to prevent stale code exploitation. To address this, MSSO implements time-limited QR lifetimes and single-use session identifiers.

From a deployment perspective, MSSO's modular design allows it to integrate into existing identity infrastructures as an 'overlay system' that complements, rather than replaces, existing Identity Providers (IdPs). This architectural flexibility makes MSSO a viable candidate for organizations looking to transition to passwordless authentication with minimal effort.

In summary, MSSO's architecture demonstrates that passwordless authentication can be implemented without compromising security. The system offers both a practical framework and a conceptual shift in how mobile and web authentication events can be securely synchronized in multi-tenant, distributed environments.

7.3 Conclusion

This thesis successfully designs, implements, and evaluates a QR-based Mobile Single Sign-On (MSSO) system. The study presents a two-stage authentication

process consisting of 'Domain Binding' for Master Token acquisition and 'Application Login' for secure Session JWT transfer. This two-stage approach provides long-term user convenience through persistent Master Tokens while guaranteeing short-term session security using disposable, ephemeral Session IDs. The MSSO architecture, which incorporates functions such as QR code-based session initiation and secure STOMP/WebSocket token push, effectively solves the common problem created by fragmented, password-dependent authentication across multiple domains and applications.

In addition to its core functional design, the system provides strong security guarantees against common attack vectors such as token interception, session hijacking, and unauthorized device access. The Master Token binding process, combined with device-specific identifiers and OTP verification during initial onboarding, ensures that tokens cannot be misused even if a device is lost or compromised. Moreover, the short-lived Session JWTs transmitted over WebSocket channels reduce the exposure window of active sessions, while allowing real-time token delivery and revocation.

From a usability and performance perspective, empirical evaluations demonstrate substantial improvements over traditional SSO mechanisms. The system achieved more than a 70% reduction in login times, while user experience studies reported a System Usability Scale (SUS) score of 84.5, indicating excellent acceptance and intuitive operation. These results highlight the MSSO system’s ability to maintain a delicate balance between security, usability, and operational efficiency, making it suitable for both enterprise and educational identity management scenarios.

Furthermore, MSSO's modular and extensible design lays the groundwork for future enhancements, including the integration of biometric authentication, adaptive session management, and hybrid fallback mechanisms for offline or proximity-based authentication. Overall, the MSSO system offers a robust, next-generation solution for identity and access management in multi-tenant and multi-domain environments, providing a practical roadmap (blueprint) for secure, scalable, and user-friendly authentication in modern digital ecosystems.

7.4 Practical further work

Master Token Lifecycle Management: Implementing features within the MSSO Management Console for users to easily view, manage, and revoke their bound mobile devices and Master Tokens.

Scalable Messaging Infrastructure: Exploring alternatives to RabbitMQ/STOMP protocols (such as AWS AppSync or Google Cloud Pub/Sub) to manage higher throughput and improve system reliability under heavy load.

Step-up Authentication: To enhance security, an optional step-up authentication mechanism is implemented that requires re-authentication for critical transactions.

Client Web SDK Monitoring: Developing real-time logging within the Client Web SDK to monitor QR code generation failures and WSS connection drops in production environments.

Biometric Protection for Mobile Application Launch: A potential improvement to further enhance the security of the MSSO Mobile Application against device theft or unauthorized access is to require biometric verification upon the first launch of the application after installation or a device reboot. This measure will guarantee that unauthorized individuals cannot access the application and the securely stored Master Tokens even if the mobile device is lost or stolen.

The proposed biometric mechanism will leverage device-specific security features such as fingerprint sensors or facial recognition and use hardware-backed storage units to protect biometric data. The authentication flow can be implemented as follows:

Upon launching the MSSO Mobile App, the user would be prompted to perform biometric verification before accessing any functionality or stored tokens.

Successful verification would unlock the app, granting access to the Master Tokens and the application interface.

Failed attempts would trigger security countermeasures, such as limiting the number of retries, logging suspicious activity, or requiring fallback authentication (e.g., PIN), maintaining a strong security posture.

This approach also lays the groundwork for adaptive security features, such as risk-based session management, remote lock or wipe in case of theft, and future passwordless authentication flows.

7.5 Theoretical Further Work

Several ways exist to further strengthen and expand the MSSO framework from both theoretical and security perspectives:

Formal Protocol Verification: Applying formal security verification methods like ProVerif or CasperFDR to the MSSO token exchange protocol can provide mathematically rigorous proofs of security features such as confidentiality, integrity, and authentication. This process will involve modeling the entire protocol, including mobile application interactions, Master Token issuance, session setup, and token transfer flows. Potential outputs include identifying subtle flaws in the protocol, making logical inferences about replay attacks, token theft, or man-in-the-middle (MITM) scenarios, and providing demonstrable resistance guarantees against well-defined adversarial patterns. This type of formal verification can also serve as a basis for compliance with industrial security standards or certifications.

Proximity Integration and Hybrid Fallback (HTE): The theoretical design and investigation of the security implications of integrating proximity-based communication channels such as Bluetooth Low Energy (BLE), NFC, or USB as a private and secure fallback mechanism could significantly reduce the mobile phone's reliance on network connectivity for critical operations. In such a hybrid scheme, the mobile device could securely exchange tokens or session information with nearby trusted devices or terminals in situations where standard online channels are unavailable. This approach would require careful threat modeling to address risks such as relay attacks, device impersonation, and eavesdropping, while ensuring the fallback channel adheres to the same strong cryptographic guarantees as the primary token exchange protocol. Furthermore, hybrid fallback can be extended to support

context-aware authentication policies such as proximity-based session activation, multi-device trust establishment, or emergency recovery mechanisms that activate when the network/server connection is interrupted.

REFERENCES

Akman, R., & Dalkilic, G. (2025). *Msso:Qr-based Passwordless Single Sign-on Architecture for Multi-Domain Applications*. 2025 18th International Conference on Information Security and Cryptology (ISCTürkiye), 1–6. https://doi.org/10.1109/ISCTrkiye68593.2025.11224825

Akman, R., & Dalkılıç, G. (2025). *Securing Health Data on Android Using Homomorphic Encryption with Microsoft SEAL*. 2025 International Conference on Smart Applications, Communications and Networking (SmartNets), 1–5. https://doi.org/10.1109/SmartNets65254.2025.11106851

Al Kabir, M. A., & Elmedany, W. (2022). *An Overview of the Present and Future of User Authentication*. 2022 4th IEEE Middle East and North Africa COMMunications Conference (MENACOMM), 10–17. https://doi.org/10.1109/MENACOMM57252.2022.9998304

Al-Haija, Q. A., & Al-Salameen, S. O. (2024). *Biometric Authentication System on Mobile Environment*: A Review. Computer Systems Science and Engineering, 0(0), 1–10. https://doi.org/10.32604/csse.2024.050846

Armando, A., Carbone, R., Compagna, L., Cuellar, J., & Tobarra, L. (2008). *Formal analysis of SAML 2.0 web browser single sign-on*. Proceedings of the 6th ACM Workshop on Formal Methods in Security Engineering, 1–10. https://doi.org/10.1145/1456396.1456397

Blundo, C., & D'Arco, P. (2005). *Analysis and Design of Distributed Key Distribution Centers*. Journal of Cryptology, 18(4), 391–414. https://doi.org/10.1007/s00145-005-0407-0

Cooijmans, T., de Ruiter, J., & Poll, E. (2014). *Analysis of Secure Key Storage Solutions on Android. Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones & Mobile Devices*, 11–20. https://doi.org/10.1145/2666620.2666627

D. Hardt, Ed. (2012). The OAuth 2.0 Authorization Framework. https://doi.org/10.17487/rfc6749

Divyabharathi D. N., & Cholli, N. G. (2020). *A Review on Identity and Access Management Server (KeyCloak). International Journal of Security and Privacy in Pervasive Computing*, 12(3), 46–53. https://doi.org/10.4018/IJSPPC.2020070104

Engineer, S., & Engineer, A. (2025). *Keeping the Data Secure. In The Structure and Interpretation of the SMB Protocol* (pp. 113–135). Springer Nature Singapore. https://doi.org/10.1007/978-981-10-7805-7_6

Fette, I., & Melnikov, A. (2011). *The WebSocket Protocol*. https://doi.org/10.17487/rfc6455

Ghorbani Lyastani, S., Schilling, M., Neumayr, M., Backes, M., & Bugiel, S. (2020). *Is FIDO2 the Kingslayer of User Authentication? A Comparative Usability Study of FIDO2 Passwordless Authentication. 2020 IEEE Symposium on Security and Privacy (SP)*, 268–285. https://doi.org/10.1109/SP40000.2020.00047

Howes, T., Smith, M., & Good, G. S. (2003*). Understanding and deploying LDAP directory services*. Addison-Wesley Professional.

Jannett, L., Westers, M., Wich, T., Mainka, C., Mayer, A., & Mladenov, V. (2024). SoK: SSO-MONITOR - *The Current State and Future Research Directions in Single Sign-on Security Measurements*. 2024 IEEE 9th European Symposium on Security and Privacy (EuroS&P), 173–192. https://doi.org/10.1109/EuroSP60621.2024.00018

Jawad Hussain, & Talib, I. (2023). *Survey Understanding Man-in-the-Middle (MITM) Attacks: Techniques, Tools, and Prevention in the Digital Era. Journal of Computing and Artificial Intelligence*, 1(2). https://doi.org/10.69591/jcai.1.2.5

Jones, M. and B. J. and S. N. (2015). Rfc 7519: *Json web token (jwt)*. IETF Trust.

Jubur, M., Shrestha, P., & Saxena, N. (2025). *An In-Depth Analysis of Password Managers and Two-Factor Authentication Tools*. ACM Computing Surveys, 57(5), 1–32. https://doi.org/10.1145/3711117

Karim, N. A., Khashan, O. A., Kanaker, H., Abdulraheem, W. K., Alshinwan, M., & Al-Banna, A.-K. (2024). *Online Banking User Authentication Methods: A Systematic Literature Review*. IEEE Access, 12, 741–757. https://doi.org/10.1109/ACCESS.2023.3346045

Kim, D., Shin, J., Ryu, G., & Choi, D. (2025). *HiPass: Hijacking CTAP in Passkey Authentication.* IEEE Access, 13, 92086–92101. https://doi.org/10.1109/ACCESS.2025.3570377

Kiourtis, A., Giannetsos, T., Menesidou, S., Mavrogiorgou, A., Symvoulidis, C., Graziani, A., Kleftakis, S., Mavrogiorgos, K., Zafeiropoulos, N., & Gkolias, C. (2023). *Identity management standards: a literature review*. Computers and Informatics, 35–36.

Kumar Barik, L. B., & Barik, N. (2021). *WhatsApp Web QR code: Auto-change Frequency Analysis*. Journal of Information Studies & Technology (JIS&T), 2021(2). https://doi.org/10.5339/jist.2021.10

Kumar, N., Jain, S., Shukla, M., & Lodha, S. (2022). *Investigating Users' Perception, Security Awareness and Cyber-Hygiene Behaviour Concerning QR Code as an Attack Vector* (pp. 506–513). https://doi.org/10.1007/978-3-031-06394-7_64

Kunke, J., Wiefling, S., Ullmann, M., & Iacono, L. Lo. (2021). *Evaluation of account recovery strategies with fido2-based passwordless authentication*. ArXiv Preprint ArXiv:2105.12477.

M. Taraka Rama Mokshagna Teja, & K. Praveen. (2022). *Prevention of Phishing Attacks Using QR Code Safe Authentication. Inventive Computation and Information Technologies*: Proceedings of ICICIT 2021, 361–372.

Mannan Javed, N. T., Khan, F. A., & Ashraf, M. (2025). *14 Access Control Techniques for Cloud Computing. Securing the Digital Realm*: Advances in Hardware and Software Security, Communication, and Forensics, 147.

Mayank, M., & Garg, M. (2019). *Developing Applications with Azure Active Directory*. Springer.

Morkonda, S. G., Chiasson, S., & van Oorschot, P. C. (2021). *Empirical Analysis and Privacy Implications in OAuth-based Single Sign-On Systems*. Proceedings of the 20th Workshop on Workshop on Privacy in the Electronic Society, 195–208. https://doi.org/10.1145/3463676.3485600

Onwuegbuzie, I. U., & Alabi, O. A. (2025). *A Review of Authentication and Authorization Mechanisms in Zero Trust Architecture*: Evolution and Efficiency. Tech-Sphere Journal for Pure and Applied Sciences, 2(1).

Ozkan, C., & Bicakci, K. (2020). *Security Analysis of Mobile Authenticator Applications*. 2020 International Conference on Information Security and Cryptology (ISCTURKEY), 18–30. https://doi.org/10.1109/ISCTURKEY51113.2020.9308020

Qatinah, S. H., & Al-Baltah, I. A. (2024). *Kerberos Protocol: Security Attacks and Solution.* 2024 1st International Conference on Emerging Technologies for Dependable Internet of Things (ICETI), 1–7. https://doi.org/10.1109/ICETI63946.2024.10777133

Rani, P., Singh, S., & Singh, K. (2024). *Cloud computing security: a taxonomy, threat detection and mitigation techniques*. International Journal of Computers and Applications, 46(5), 348–361. https://doi.org/10.1080/1206212X.2024.2319937

Soltani, R., Nguyen, U. T., & An, A. (2021). *A Survey of Self-Sovereign Identity Ecosystem*. Security and Communication Networks, 2021, 1–26. https://doi.org/10.1155/2021/8873429

White, M. D., & Oostrom, M. (2003). *STOMP Subsurface Transport Over Multiple Phases* Version 3.0 User's Guide.

Wilson, Y., & Hingnikar, A. (2023). *Solving identity management in modern applications: demystifying OAuth 2.0*, OpenID Connect, and SAML 2.0.

Yerram, S. (2024). *Okta Identity and Access Management. Journal of Advances in Developmental Research*, 15(1). https://doi.org/10.71097/IJAIDR.v15.i1.1575

Yusop, M. I. M., Kamarudin, N. H., Suhaimi, N. H. S., & Hasan, M. K. (2025). *Advancing Passwordless Authentication: A Systematic Review of Methods, Challenges, and Future Directions for Secure User Identity*. IEEE Access, 13, 13919–13943. https://doi.org/10.1109/ACCESS.2025.3528960

Zineddine, A., Belfaik, Y., Rehaimi, A., Sadqi, Y., & Safi, S. (2025). *Single Sign-On Security and Privacy: A Systematic Literature Review*. Computers, Materials & Continua, 84(3), 4019–4054. https://doi.org/10.32604/cmc.2025.066139